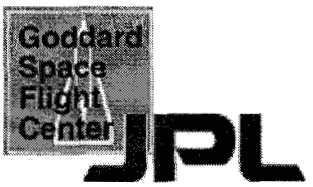
JPL D-19449

2nd GSFC/JPL Quality Mission Software Workshop Report Held May 16-18, 2000, San Diego, California



July 30, 2000

Prepared by:

C. Y, Lin
Jet Propulsion Laboratory

Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Contents

Section **Page**

Tables

The following individuals have made significant contributions to this report:

Rich Doyle (JPL)

Scott Green GSFC)

Trisha Jansma (JPL)

Howard Kea (GSFC)

John Kelly (JPL)

Roger Lee (JPL)

Pat Liggett (JPL)

Carman Mikuski (JPL)

Dave Nichols (JPL)

Ken Rehm (GSFC)

Mike Stark (GSFC)

1. Introduction

The 2nd GSFC-JPL annual Quality Mission Software (QMSW) Workshop took place in Fallbrook, CA on May 16-18, following the 1st Workshop held in Annapolis, MD in August 1999. The theme of the August workshop was mostly information exchange, while the theme of the 2nd workshop was seeking collaboration opportunities. In both workshops, it became immediately apparent that NASA's two science mission centers share common viewpoints on issues, goals and lessons learned in producing quality mission software.

The 2nd GSFC-JPL QMSW workshop brought together 56 participants mostly from GSFC and JPL to focus on critical challenges for mission software. Representatives from ARC, JSC and NASA IV&V also participated. The meeting was co-chaired by Marti Szczur, former Chief of the Information Systems Center at GSFC, and Richard Doyle, Division Manager for Information Technologies and Software Systems and Leader of the Center for Space Mission Information and Software Systems at JPL.

1.1 Purpose

The purpose of the Fallbrook workshop, in addition to technical information exchange, was to seek collaboration opportunities in developing solutions to meet common challenges in producing quality software for NASA science missions.

Participants who attended the workshop represent the communities of project and task management, software engineering, software technology development, and software processes. The topics of the workshop focused on five main themes:

- Concerns of Software Practitioners
- Future Infrastructure and Technology for Software Development
- Software Metrics
- Relationships to Projects
- Mission Software Sub-disciplines

1.2 Organization of This Report

This report includes the following sections:

- Section 1 contains introductory material
- Section 2 describes each session's objectives, and provides a brief summary and recommendations
- Section 3 includes a workshop summary
- Section 4 contains workshop participant information
- Section 5 is a list of acronyms

2. Objectives and Summaries of Workshop Sessions

This section describes objectives, provides brief summaries, key findings, recommendations and collaborative opportunities for each workshop session.

2.1 Software Practitioner Concerns

The objectives of the Software Practitioner Concerns session were to identify and gain a better understanding of issues related to IT practitioner concerns and ongoing activities related to the IT workforce at GSFC and JPL.

Co-Chairs: Scott Green and Trisha Jansma

2.1.1 Session Summary

This session was broken into two parts: an initial set of four presentations designed to share information regarding existing activities related to IT workforce issues, followed by an open discussion around general IT practitioner concerns.

Part One - Presentations

The topics covered in the session included:

- CSMISS IT Workforce Enrichment Element
- GSFC STAAC/AETD Career Development Program
- NASA Software Working Group Training Subcommittee
- NASA IT Workforce Challenge Team

The CSMISS IT Workforce Enrichment Element presentation focused on the IT Workforce Enrichment Element of JPL's Center for Space Mission Information and Software Systems (CSMISS). The primary objectives of this element are to:

- Build a sense of community among IT professionals
- Provide career growth and technical skill enhancement
- Provide a voice to address issues of mutual concern to IT professionals and their managers

The GSFC STAAC/AETD Career Development Program presentation involved a discussion on an ongoing effort between GSFC Systems, Technology and Advanced Concepts (STAAC) and Applied Engineering Technology (AET) Directorates to design and develop a web-based career development and planning tool. The tool will serve to provide engineering professionals with resources to view career planning from two vantagepoints:

- A career track, which highlights the progression of job positions and the skills, knowledge and training associated with each
- A career planning roadmap, which highlights activities for employees to undertake when examining their career options

The NASA Software Working Group Training Subcommittee presentation focused on the NASA SWG Training Subcommittee chaired by John Kelly of JPL. Objectives of the committee are to:

- Maintain agency capabilities in software technology
- Establish a core set of training requirements for NASA software practitioners and managers
- Review and development associated courses
- Interface with other groups providing software training within and outside the agency

Finally, the NASA IT Workforce Challenge Team presentation provided an overview of this agency-wide team to address IT workforce issues. The team was established in May 1999 as part of a proactive, long-term strategy to develop an integrated training and development program that will enable NASA to meet its long-term IT goals. Three workshops have been held to date, and the following five subcommittees have been formed:

- Recruitment and Retention
- IT Training
- CIO Development Model
- IT Skill Survey
- IT POP Analysis

Part Two - Open Discussion

An open discussion followed the presentations and focused on several topics related to IT concerns. Among the topics discussed were:

- The impact of current government practices and bureaucracy
- How to better recruit potential IT personnel
- Retention problems and experiences at GSFC and JPL
- Why IT personnel leave the NASA workforce
- The benefits of working for NASA vs. the private sector
- How practitioners limit themselves in career development
- Workforce diversity
- Clarity in roles and responsibilities of software architects, software managers, software system engineers, and systems engineers

2.1.2 Key Findings, Recommendations and Collaboration Opportunities

The following areas of possible collaboration opportunities were identified:

- Curriculum development for and participation in these classes
  - Understanding Software for Project Management
  - Software Architect Program
  - Understanding Missions for Software Developers
- GSFC to participate in (conjoin) the JPL IT Spotlight and IT Seminar Series
- JPL to participate in a SEL Workshop with a three-hour tutorial on Software for Project Management
- Define roles and responsibilities for Software Architect, Software Manager and Software System engineers
- GSFC ISC to become a more proactive participant in providing input to NASA IT Workforce Challenge activities
- Explore potential synergy amongst the NASA Software Working Group Training Subgroup, the JPL training activities, the NASA IT Workforce Challenge Training Subgroup, and NASA Code FT regarding Curriculum development, training sessions, competency level targets, etc.
- Develop points-of-contact and timeline for IT practitioner activities

2.2 *Future Infrastructure and Technology for Software Development*

The objective of this session was to present and discuss middleware as used in the development process and the technology and techniques for integrating distributed and disparate systems. A goal is to develop joint proposals for development of middleware capabilities for the purpose of developing collaborative systems.

Co-Chairs: Howard Kea and Patricia Liggett

2.2.1 *Session Summary*

The goals of the NASA Faster-Better-Cheaper focus and current center directives are calling for reduced budgets. To achieve this, NASA Enterprises and Centers are developing shared projects and programs that fit within reduced budgets and are seeking to leverage the expertise at the centers to make better, cost effective use of existing talent. In addition, since many of the Code S and Code Y technology initiatives are often very similar and have the potential for shared development, the idea of developing middleware applications to increase reuse, sharing and collaboration is timely and necessary.

The topics presented in this session were:

- An Overview of State-of-the-Practice/Art for Middleware
- Middleware Tools (Part 1)
- Middleware Tools (Part 2)
- ISE Mars Application
- Automated Software Verification

Five presentations on technology were given. Norm Lamarra (JPL) presented an overview of middleware standards such as HLA, COM, XML, Corba, Java, etc. This presentation was followed by two presentations on development using XML to support distributed user access in the Troy Ames (GSFC) application for instrument operations and for distributed data set access in the Dan Crichton (JPL) application. Both addressed the issues of distributed and disparate users and a solution that allows interaction and integration. Also, via video conferencing from GSFC, Eric De Jong (JPL) presented the Intelligent Synthesis Environment (ISE) for the Mars Exploration application. The ISE vision and long-term goal, as well as a fifteen-year program roadmap to achieve this vision, were described in De Jong's presentation. Lastly, Klaus Haverland (ARC) presented an overview of the research activities at ARC on an automated software verification technique. Haverland illustrated the verification technique by presenting a case study of its application to the Deep Space 1 Project.

2.2.2 *Key Findings, Recommendations and Collaboration Opportunities*

The result of these presentations and follow-on discussions was to proceed with further discussions and workshops on issues related to middleware and collaborative systems development and to pursue joint proposals for upcoming research announcements. Areas identified for collaboration include:

- Plans to jointly propose programmatic initiatives at the NASA level (e.g., a new software emphasis in the ISE program)
- Exploring applications of the GSFC-developed IML (Instrument Modeling Language) at JPL
- Conduct middleware workshops. Proposed sessions are:
  - Practitioner
  - Mission marketing
  - Middleware working group to address the capabilities of XML, HLA, etc.
  - Middleware for software program initiatives

2.3 *Metrics*

The objectives of this session were to: a) Examine metrics from the viewpoints of the goals of practitioners and projects in the context of GSFC/JPL science missions; b) capture the interests of these users and the goals they want to accomplish by using metrics; c) capture what the current obstacles are to the widespread use of metrics and develop strategies for overcoming these obstacles; d) capture project goals from different perspectives that the SEL and MSP groups could carry forward into developing a useful metrics program; and e) evaluate the NASA Core Metrics against these goals to determine the next steps for the SEL and MSP groups to carry forward metrics work.

Co-Chairs: John Kelly and Mike Stark

2.3.1 *Session Summary*

The definition and acceptance of a metrics set for software acquisition and development is not a trivial undertaking. At GSFC, the Software Engineering Laboratory (SEL) and Software Assurance Technology Center (SATC) have been working for over a year to identify a set of core software metrics. The core metrics are intended to represent a minimal set that should be collected by projects to monitor important factors in the health of both the software product and development process. At JPL the Mission Software Process (MSP) task has been working on an initial set of software metrics to apply uniformly to all JPL Flight projects. JPL has focused on tailoring both the Core Metrics set and the Practical Software Measurement (PSM) Program (originated under the Office of the Under Secretary of Defense). The purpose of this workshop was to bring these two groups together in an environment of Goddard and JPL software developers, testers, QA personnel, Cognizant engineers, and managers to discuss

software metrics issues. This information will greatly aid in the tailoring of a metrics set that could be applied across similar projects.

Since one of the main goals of this session was to get feedback for tailoring an effective set of software metrics, thus the session was conducted in a workshop format with three hours equally divided between presentation and discussion. The Goal/Question/Metric (GQM) paradigm was used as a model for designing the presentations and exercises in this session. The original agenda for this session included the following:

- Presentation 1 – Who Measures and Why?
  - Includes overview of Goal/Question/Metric (GQM) paradigm
- Exercise 1 – Goal Setting
  - List Project and Software Practitioner roles that are suppliers or consumers of software metrics
  - For each of the identified role categories, define the software business goals.
- Presentation 2 – Strategies for Metrics Infusion
- Exercise 2 – Identify obstacles and strategies to overcome them
- Presentation 3 – Choosing from available metrics sets (NPD 2820.1, JPL D-15378, NASA Core Metrics, and PSM)
- Exercise 3 – Prioritize a subset of the NASA Core Metrics that matches the goals from Exercise 1

The agenda was modified both before and during the session. Exercise 2 was removed to allow time for discussion of the IV & V Facility management planning in advance of the session, and the agenda was modified during the session to include open discussion of the issues concerning software metrics on projects.

Exercises One

The main objectives of this exercise were to examine project and practitioner goals. Identifying these goals allows the selection of metrics that add value to software development projects. Exercise three provides some initial priorities for data collection; and all the information collected here can inform MSP and SEL personnel in defining, collecting, and reporting useful metrics to projects and practitioners.

In the two parts of exercise 1 the participants supplied two sets of information: 1) the roles associated with the project and practitioner perspectives, and 2) the business goals associated with each perspective. The process perspective was not presented due to time constraints. The data collected is provided below in *Italics:*

PROJECT – Roles

1. *Software Manager / Software Element Manager*
2. *Software Architect*
3. *Integration Manager*
4. *Operations Manager*

5. *Mission Assurance Manager*
6. *Program Manager*
7. *Mission System Engineer*
8. *Subsystem Lead*
9. *Software Verification Lead*

PRACTITIONER – Roles

1. *Team Lead*
2. *Subsystem Lead*
3. *Software Developer*
4. *Configuration Management Coordinator*
5. *Tester*
6. *Development Environment Lead (Testbed)*
7. *IV&V / QA*
8. *Software "Toolsmith"*
9. *Line Manager*
10. *Technologist (R & D)*
11. *Sustaining Engineer*

The following are the Business Goals associated with each perspective provided by the workshop participants in Exercise 1.

PROJECT - Business Goal

1. *On-time Software Delivery*
2. *Within Budget*
3. *Meet Mission Requirements*
4. *Accurate Estimation (job scope)*
5. *Flight Software / No Failures*
6. *Operability*
7. *Maintainability of Software*
8. *Life Cycle Cost*
9. *"Inheritability" of Software / Reuse*

PRACTITIONER - Business Goal

10. End-to-end Software Reliability

11. On Time / Adequate Resources

12. Low Complexity of Code

13. Stability of Requirements / Ease of Managing Instability

14. Meet Performance Requirements / Resource Obligations

15. Testability / Coverage

16. Development Environment / Input into Selection Process

17. Re-planning / Voice in Re-planning Due to Change (Requests)

18. Good Architecture (Existence / Quality)

19. Timely Planning /Prep of Testbed / Test Environment

20. Adequate Testing

21. Insertion of new Technologies

22. Adequate training

23. "Have a Life"

24. Mission Success

25. Recognition of Contribution

26. Clarity of Project Plan / Task Plan

Time for this workshop did not allow for each goal to be analyzed or broken down into a set of derived questions per the Goal/Question/Metric (GQM) paradigm. However, useful insights could be gained by examining major categories of inputs that were provided by the participants across the underlying subcategories of project cost, schedule performance and product quality.

Observations Related to the Project Goals

While it was useful to identify the roles, we did not have time to work on goals for each individual role. However, they were still useful to have on paper while considering goals for a given perspective. The project goals can be grouped into two major categories: project cost and schedule performance and product quality. Project performance includes goals 1,2,4, and 8; product quality the remaining goals. It was not clarified at the workshop whether "life cycle cost" (goal 8) was intended to mean end-to-end development cost or to include maintenance and operations as well.

In grouping project performance goals, one can further observe that accurate estimation is a necessary condition for the other goals of on-time delivery and controlling costs. It is not sufficient for on-time delivery, as the estimated resource needs may not be met. However, it is probably the most important of these goals, so metrics should be used to assess and improve estimation accuracy.

The five remaining product quality goals can be divided into two subgroups. The first contains the goals of meeting requirements and avoiding *any* flight software failures. The second contains the goals of operability (usability), maintainability, and "inheritability" (reusability). The first subgroup, particularly avoiding mission failures, is clearly more important to project managers. In the words of one project manager, "This [flight software failure] is what keeps me awake at night". None of the other four quality concerns were nearly as important in the discussion, although there was a sense that "meeting mission requirements" was a given.

Observations between input categories of Practitioners Goals and Project Goals

- In general, most of the Practitioners' Goals also address Project Goals. Some of them are fairly direct links showing good alignment (e.g., goal #24 "mission success" relates to both #3 "meet mission requirements" and #5 "flight Software with no failures"). However, others are more derivative (e.g., goal #12 "low complexity of code" relates to both #7 "maintainability" and #9 "Inheritability of Software / Reuse")
- Some Practitioners' Goals do not seem to have a direct match to the Project's Goals (e.g., #16 "development environment selection", #21 "inserting new technology", #23 "have a life" and #25 "Recognition of Contribution"). This does not mean they are not important at the project level, just that a direct relationship is difficult to make. The only goals that seem to invite conflict are the Practitioner Goal of #21 "inserting new technology" and the Project Goals of #1 "on-time software delivery", #2 "within budget", and #5 "flight Software with no failures".

Further probes into these questions of goal alignment and conflict between project and practitioners would be both interesting and insightful. Underlying conflicts in both goals and communication could result in software products having a negative impact on the end system.

Exercise Three – Prioritize Using the NASA Core Metrics to Match the Business Goal from Exercise 1

The purpose of this exercise was to:

- Capture the metrics indicators and goals that the participants want to accomplish by using a particular indicator
- Match the goals to the indicators that projects and practitioners could use to gain visibility into true progress
- Compare the NASA Core Metrics against project and practitioner goals to help JPL and Goddard metrics advocates tailor an effective set to implement on a broader scale
- Determine how important the different metrics indicators are for managers and developers

The group was given a table with a list of the NASA Core Metrics indicators (see Appendix A) and were asked to:

- Provide the goal number (generated from previous exercises) for the indicators that could be used to determine if that particular goal is being achieved
- Prioritize the NASA Core Metrics indicators (from 0 to 3, 3 being the highest)
- Add additional indicators that were not on the original NASA Core Metrics indicators list, but are important to meet their business goals

During this exercise the participants were divided into two groups, each with three teams. One group was representing the managers' point of view and, the other representing the practitioners'. All feedback (worksheets) from the teams was collected for a follow-up analysis. Table 1 summarizes the average priority rating for the managers' teams compared to that of the developers' teams, and the average for all teams. Indicators are ordered by the average priorities for all teams in descending order. *Italics* represent data provided by workshop participants.

Table 1. Average Priority Rating Comparisons

	Indicators from The NASA Core Metrics	Average Priority Managers	Average Priority Practitioners	Total Average Priority
1.	Milestone planned and actual vs. time	*3.0*	*3.0*	***3.0***
2.	Cumulative number of open and closed defects by severity over time	*3.0*	*3.0*	***3.0***
3.	Cumulative total number of open and closed defects over time	*2.6*	*3.0*	***2.8***
4.	Total source code size (estimated & actual) vs. time	*3.0*	*2.0*	***2.5***
5.	Requirement count vs. time	*2.3*	*2.6*	*2.5*
6.	Number of modifications opened and closed (defined by program/project)	*2.0*	*3.0*	***2.5***
7.	Requirement verified vs. time	*3.0*	*2.0*	***2.4***
8.	Number of TBDs, TBSs, etc.	*1.5*	*3.0*	***2.3***
9.	Total (planned and actual) hours vs. time	*3.0*	*1.3*	***2.2***
10.	Number of modifications opened and closed vs. component/doc	*2.0*	*2.0*	***2.0***
11.	Work element completion date planned and actual vs. time	*1.6*	*2.5*	***2.0***
12.	Requirement errors vs. time	*1.6*	*2.5*	***2.0***
13.	Periodic effort by phase or activity or person type (planned or actual) vs. time	*2.0*	*2.0*	***2.0***
14.	Effort staffed hours to perform modifications vs. cause	*0.5*	*2.5*	***1.5***
15.	Effort staffed hours to perform modifications vs. component	*1.0*	*2.0*	***1.5***
16.	Developed source code library size actual vs. time	*1.0*	*2.0*	***1.5***
17.	Relative complexity by component	*1.0*	*2.0*	***1.5***
18.	Number of optional phrases	*1.0*	*1.5*	***1.2***
19.	Cyclomatic complexity by component	*0.5*	*1.6*	***1.2***
20.	Requirements satisfied by OTS (estimated)	*0.6*	*1.5*	***1.0***
21.	Requirements satisfied by OTS (actual)	*0.6*	*1.5*	***1.0***
22.	Number of ambiguous phrases	*0.6*	*1.5*	***1.0***
23.	Calling complexity by component	*0.5*	*1.0*	***0.8***

The following are additional indicators suggested by the teams:

- Mission Complexity
- Operability
- Maintainability
- Resource
- Earned Value
- Product Line
- Cost
- Reuse / Inheritance
- Life Cycle
- Requirements Change Tracking
- Requirements vs. Plan

The workshop participants gave *very high priorities* to measurements of software development progress (milestones) and progress toward closing out defects. These are fundamental measures that indicate whether the software development effort is converging toward a quality product within the context of the overall project schedule. These measures provide "flags" to determine whether management intervention may be needed to re-balance resources verses schedule. At the *bottom* of the metrics priorities provided by the participants were measures of complexity, Off-The-Shelf (OTS) software, and the counts of "ambiguous" and "optional" requirements phrases. Although these may have importance in cause-and-effect relationships with other metrics, they do not appear to be fundamental indicators of software development health. In a minimal software metrics collection program, these may be best listed as optional.

There is an interesting disparity in the priorities assigned by the Project Management verses Practitioner groups on many of the metrics. Ten of the 23 metrics received at least a 1.0 or greater average priority difference between these two groups (on a 3-point scale). Many of these ten metrics appear in the middle portion of the above table. This difference between priorities between these two groups could be significant. A more controlled survey may be warranted to determine if this difference is real. If these groups have different priorities for a sizable portion metrics sets, it could explain previous difficulties in implementing software metrics on projects. One of the root causes of these difficulties could be that neither group fully "bought into" the selected software metrics set.

It should be noted that the workshop teams had a very short time (30 minutes) to reflect on the second part of this exercise: mapping the business goals to the corresponding metrics. Because of time constraints, the data collected is partially incomplete and hasn't been analyzed for this study.

2.3.2 *Key Findings, Recommendations and Collaboration Opportunities*

During the workshop, a request was made to have a segment added for open discussions on issues concerning projects and software metrics. This was suggested to ensure that concerns were covered that may not have been anticipated and to allow an open exchange between the project manager's present and practitioner representatives from the GSFC and JPL software communities. The key issues discussed are provided below. The data collected from this discussion has only been altered for readability. The notes taken on the flip charts are shown in *italics*. The bullets provided by the participants were grouped into four areas by the session co-chairs.

Comment Area #1: Metrics Related to Communication on Software Development Status

- *Projects need an "early flag" to indicated when software needs attention*
- *Could the software team get to where they need to be from where they are now?*
- *Statement: Metrics concerning the complexity of code is really a software developer issue, rather than a project issue. Response: There is a correlation between complexity metrics and fault history (it also has an influence upon the test time)*
- *Metrics concerning performance, memory and CPU load need to be visible and managed at the project level*
- *Test metrics are needed: number of software tests to be run versus progress (number successfully passed)*
- *Projects are listening to software concerns better than they did in the past*
- *Metrics needs to be "rolled up" to be properly monitored at the project level*
- *Milestones for software development need to be relatively small (two-month increments for a three-year project) to monitor progress*

Comment Area #2: Reliability of Software

- *We are working in an environment where we can't tolerate a single mistake*
- *Projects need certainty against error during operation/flight*
- *Early indicators of reliability are needed to effectively adjust resources and implement mitigation. Late indicators of software reliability are very problematic for projects*
- *The software team needs to have a good handle on Fault Protection during development*
- *Question: How do you know the software will not have failures? Answer: Review quality, Key people in reviews, Code Reviews, Testing, and Fidelity of the testbed, ...*

Comment Area #3: Requirements and Software

- *Requirements are not carved in stone. Risks drive the requirements. Risk management is a key goal of project managers*
- *Requirements need to be mapped out by builds. The metrics that are needed by build are: 1) Number of requirements changed/added, and 2) Number of requirements tested by phases of development*
- *One of the big questions is whether the project is in trouble because of requirement changes. Factors include the number of changes over time and the impact of the changes*
- *Late changes to requirements affecting software are particularly problematic. A process needs to be implemented which re-analyzes requirements when late requirements changes occur*
- *Configuration Management control should help solve requirements problems, but theory and practice could be very different*
- *Requirements changes could come from many sources*
- *Software teams have "pushed back" on excessive requirements changes during projects*
- *Test personnel are the ones who really get caught in a bind with requirements changes*

Comment Area #4: Estimates for Software Development

- *How much testing is needed on a project and how does it get accurately estimated*
- *Issues involving tools for software estimating: Grow estimation tools with the projects (home grown). Tools need to be tuned to give accurate estimates (COCOMO)*
- *An early understanding of how much is it going to take to do a job is critically needed (number of testbeds, size, etc.)*
- *Estimates of allocated memory and performance by function are needed*
- *Reliable estimates of tests and the number of hours it will take to conduct them is needed*
- *Estimates from software principals could be* verifiable *(example: 7 principals on a project, but only 3 provide trustworthy estimates)*
- *Useful metrics usually come from previous projects and the current project's schedule history*

The above-summarized statements from the participants largely speak for themselves. However, there are a couple of observations that are worth noting:

- The statement "We are working in an environment where we can't tolerate a single mistake" is a very strong one, but it is consistent with the Business Goals concerned with avoiding software failure (#5, #24) stated in the previous section and the priorities set for metrics in the next section.
- Several statements point to the need for metrics to be visible, timely, and interpretable by Project Managers as well as produced in a way that timely corrective actions could be taken. Examples include, "Projects need an 'early flag'...", "Early indicators of reliability are needed...", and "One of the big questions is whether the project is in trouble because of requirement changes."

Software process improvement groups (SEL at Goddard and MSP at JPL) and assurance organizations (SATC at Goddard and SQA at JPL) piloting and deploying metrics need to address these observations. Their tailoring of metrics to meet the needs of project managers and software development teams is enhanced by this discussion.

In Summary, the workshop conducted within this session was highly productive in identifying priorities for metrics and in improving how this information may be collected via facilitated workshops.

One crucial conclusion that could be drawn from this workshop is that not all metrics are of equal value to projects and practitioners. However, there was a clear and strong consensus that measuring defects was of the highest priority. Other issues cited as being important from the project point of view were:

- Requirements stability
- Cost and schedule estimation
- Tracking test progress
- CPU and memory usage

Recommendations Identified for This Session

There were two main lessons learned from the Metrics session workshop. First, GQM exercises probably work better for a small group of people. Second, the mapping of goals to roles within the practitioner, project, or process communities is probably a workshop in itself.

Additional metrics workshops should be planned to provide additional data points to validate the conclusions. Some thoughts for consideration in future workshops are:

- More time should be allocated to the exercises
- Individuals filling in the form and providing their role on the project may provide a good way to validate the results

- Ask the participants to provide additional information, such as:
  - Describe an additional goal (outside the provided list) for an indicator
  - Describe reasons why an indicator is of no use for them
  - State if they have experience using the indicators, collecting the metrics or analyzing them

Finally, the metrics session co-chairs should collaborate closely with the NASA Software Working Group and the Center Software Engineering Process Group. The information developed during this workshop should be supplied to these two groups for developing a recommended set of metrics and how to use them.

2.4 *Relationships with Projects*

The objective of this session was to engage a small subset of flight project managers in a discussion of software engineering issues to better understand their perspectives, needs, and priorities.

Co-Chairs: Dave Nichols and Elaine Shell

2.4.1 *Session Summary*

The JPL and GSFC software engineering community has been developing a number of products intended to support project managers in developing robust, error-free software on time and within budget. There were six project managers in attendance. Five were physically present and one participated via video-conferencing. The flight project mangers were Dave Gallagher (SIRTF), Avi Karnik (AIRS), Dolly Perkin (EOSDIS), Chet Sasaki (GENESIS), Leslie Livesay (Deep Space 3) and Harry McCain (Polar Orbiting Environmental Satellites – POES).

The topics covered in this session were:

- Overview of Software Classes for Project Managers
- Software Quality Assurance
- Software Development Principles
- The Software IV&V Center
- Approaches and Technologies for Flight Software V&V
- Report on Metrics Discussion Session

The proposal for a software class for project managers was supported but did not engender a lot of discussion. The suggestion was made that the class should include a segment on software acquisition. GSFC would like to send a representative to participate in the first class.

The software quality assurance presentation involved a discussion of the SQA policy that JPL has put in place. The policy describes the role of the JPL Software QA organization, the Project, and the NASA IV&V Center. Specifically, it gives the Software QA organization the responsibility for making a software criticality assessment and a recommendation to the project for software assurance activities. The project must make its own decisions on SQA practices, but if an IV&V program is planned, the software QA organization will provide the NASA IV&V Center business-level interface.

Ted Hammer discussed the transfer of management of the West Virginia IV&V Center from NASA Ames to GSFC. The current policy is that all NASA projects employing IV&V will use the IV&V Center. GSFC is in the process of developing a new management plan that will go to the NASA Administrator by the end of May. The project managers were interested in any modification to the role that the IV&V center will be playing and how to get the best return on their IV&V investment.

The software principles that were described are meant to be a collection of best practices that incorporate lessons-learned in flight-critical software development. These are not requirements, but intended for use as a checklist of things that should be considered when planning, developing and testing software. The project managers were interested in the integration of the software principles over the full life cycle. That is, compliance with the Principles was not to be viewed as single event but should be evaluated throughout the entire life cycle. It was felt that conformance to software principles was a project responsibility and not an IV&V function. GSFC would like to receive a baselined version of the JPL Principles and may incorporate them into the GSFC processes similar to JPL's plans.

The presentation on verification and validation was of great interest to the project managers. One PM stated that software verification is the single most concern he had about software. The distinction between verification and validation was discussed. Verification is the testing and assurance that detailed requirements are being met. Validation involves scenario-based testing against high-level mission objectives, which, in a sense, is verifying not only the software implementation but also the requirements. Stress testing was discussed and two views emerged: 1) stress testing is simply saturating the computing resources such as memory, CPU and communications paths; and 2) stress testing involves exploring pathological boundary conditions, and resilience to unexpected inputs. The estimate that approximately 80% of a GSFC V&V effort was validation and 20% verification seemed to be enlightening.

2.4.2 *Key Findings, Recommendations and Collaboration Opportunities*

In the discussion session, the project managers identified the following additional items (care-abouts) as high priority concerns:

- Testbeds — How are testbeds validated? The validation process needs to be planned. Define the various levels of testbed fidelity. It is difficult to unambiguously specify testbed simulators (device models?) in procurement specifications. Some kind of taxonomy is needed
- Metrics — Earned value is a key metric that is required. How best to manage a software-based earned value metric
- Testing — Different types of software testing need to be more clearly specified

Possible Collaboration Identified for This Session

- Propose a NASA initiative on verification and validation
- Define Taxonomy on software simulators
- GSFC project managers participation on the JPL designed: Understanding Software for Project Management"

2.5 *Mission Software Sub-disciplines*

The objective of the Mission Software Sub-disciplines session was to identify major issues that are common between JPL and GSFC as well as common to sub-disciplines of mission software. The sub-disciplines considered were flight, real-time ground, planning and scheduling, and spacecraft trending and analysis.

Co-Chairs: Roger Lee and Ken Rehm

2.5.1 *Session Summary*

This session was conducted in a workshop format. Approximately one third of the time was spent on reporting the results of a set of surveys conducted prior to the workshop at both centers for four mission software sub-disciplines: flight software, ground software, planning and scheduling software, spacecraft trending and analysis software. The other two thirds time was used for breakout discussions.

The agenda for this session included:

- Introduction and Rules
- Report on survey results
- Break-out sessions

The survey consisted of 72 questions covering all phases of the software life cycle. There were over twelve respondents representing all of the sub-disciplines as well as the two NASA centers. After a presentation of the survey results that included a discussion on how mission software sub-disciplines are similar and dissimilar to the conference participants , the attendees were separated into two working groups (managers and engineers) to evaluate the results and make recommendations for action.

2.5.2 Key Findings, Recommendations and Collaboration Opportunities

The survey showed significant similarities in findings across centers and across sub-disciplines. Common findings included the following:

- The requirements process is becoming more informal over time. Requirements are sometimes captured only in e-mail messages or by word-of-mouth.
- Requirements are sometimes carried over from previous developments even though they have lost their relevance.
- Reviews are becoming less rigorous and less frequent.
- The capabilities of COTS software are exaggerated. The amount of time required to integrate a COTS package into a deliverable is often comparable to the time it would take to develop the same capability from scratch.
- The implementation of "faster, better, cheaper" has resulted in significant schedule pressure. Development and testing is postponed, sometimes until after launch.
- The test environment does not reproduce the operating environment with high fidelity. This is particularly true for flight software development that must be tested on the flight hardware.
- Both centers acknowledged that it is difficult to recruit and retain experience software engineers in the current environment. However, this concern was voiced more strongly by JPL respondents than by Goddard.

Overall, there is a sense that the software process has deteriorated over time. Many responses to the questions took the form "we used to follow a process in this area, but we don't anymore because we are not given the time". A related theme dealt with the overall quality of software products – engineers felt that they could not take pride in their work due to curtailed development cycles.

Recommendations Identified for This Session

- Project managers at both Centers should be trained in the fundamentals of software management. There was interest on the part of the Goddard contingent to participate in the pilot of the "Software for Project Management" course.
- The two centers should make personnel available to participate in each other's reviews. Since many of the software products and processes are similar at the two Centers, a fruitful exchange of ideas should emerge from such exchanges.
- JPL and Goddard should share their knowledge of COTS software products. Since much of the development activities of the two Centers are similar, many of the same development tools and software packages are relevant, and the experience of one Center would be useful to other in determining whether to acquire the same tool at the other Center. A simple web site would be sufficient to start this exchange.
- Redo the survey with questions focused on what is right about the software processes at the two Centers. This would highlight good software practices which could be applied across sub-disciplines and across Centers.
- Maintain the team that was formed to present this session in order to continue the exchange of information about software processes across sub-disciplines and across Centers.

3. *Summary*

The workshop concluded with an introspective post-mortem where there was clear consensus that the workshop was producing useful products and several collaboration opportunities. The beginnings of plans for a 3rd GSFC-JPL Quality Mission Software workshop ensued. The next workshop may focus on a single theme, e.g., Software Verification & Validation. March-April 2001 is the likely timeframe, and the venue will be on the East Coast, possibly Williamsburg, VA.

4. *Acronyms*

AETD	Applied Engineering Technology Directorates
AIRS	Atmospheric Infrared Sounder Project
IT	Information Technology
CIO	Chief Information Officer
COCOMO	Constructive Cost Model
COM	Common Object Model
COTS	Commercial Off-the-Shelf
CPU	Central Processing Unit
CSMISS	Center for Space Mission Information and Software Systems
GQM	Goal/Question/Metric
EOSDIS	Earth Observing System Data Information System
GSFC	Goddard Space Flight Center
HLA	High-Level Architecture
IML	Instrument Modeling Language
ISE	Intelligent Synthesis Environment
IV&V	Independent Verification and Validation
MSP	Mission Software Process
OTS	Off-the-Shelf
POES	Polar Orbiting Environmental Satellites
POP	Program Operating Plan
PSM	Practical Software Measurement
QA	Quality Assurance
QMSW	Quality Mission Software Workshop
SEL	Software Engineering Laboratory
SLOC	Source Line of Code
SIRTF	Space Infrared TBD
STAAC	Systems, Technology and Advanced Concepts
SQA	Software Quality Assurance
SWG	Software Working Group
XML	Extensible Markup Language

5. *Workshop Participants*

Wafa Aldiwan, JPL
Abdullah Aljabri, JPL
Troy Ames, GSFC
Matt Barry, GSFC
Maureen Bartholomew, GSFC
Kris Brown, GSFSC
Magdi Carlton, JPL
Daniel Crichton, JPL
Steve Condon, CSC
Rich Day, GSFC
Eric M. De Jong, JPL
Richard Doyle, JPL
Dan Dvorak, JPL
Mary Ann Esfandiari, GSFC
Anne Elson, JPL
Tina Fredo, GSFC
Dave Gallagher, JPL
Scott Green, GSFC
John Hackney, JPL
Ted Hammer, JPL
Klaus Havelund, ARC
Paul Hempel, GSFC
Paul Hunter, GSFC
Trisha Jansma, JPL
Av. Karnik, JPL
Howard Kea, GSFC
John Kelly, JPL
Frank Kuykendall, JPL
Norm Lamarra, JPL
Linda Landis, CSC
Annette Larson, JPL
Milton Lavin, JPL
Meemong Lee, JPL
Roger Lee, JPL
Pat Liggett, JPL
Chi Lin, JPL
Leslie Livesay, JPL
Harry McCain, GSFC
David McComas, GSFC
Carmen Mikulski, JPL
Ron Morillo, JPL
Dave Nichols, JPL
Dolly Perkins, GSFC
Ken Rehm, GSFC
Tom Renfrow, JPL
Linda Rosenberg. GSFC
Ioaha Rus, GSFSC
Chet Sasaki, JPL
Lisa Shears, JPL
Elaine M. Shell, GSFC
Burt Sigal, JPL
Kim Simpson, JPL
Mike Stark, GSFC
Jody Steinbacher, JPL
Sandy Steinberg, CSC
Mari Sykes, CSC
Marti Szczur, GSFC
Sandi Thomas, JPL
Mike Tilley, JPL
Mark Walther, GSFC
Marv Zelkowitz , GSF

Appendix A: Indicator Selection Scorecard

Indicators Organized By NASA Core Metrics	Comments/Your Preferences for Indicators	NASA Core
Schedule		
1) Milestone planned and actual vs. time		(P)
2) Work element completion date planned and actual vs. time		(P)
Effort		
3) Total (planned and actual) hours vs. time		(P)
4) Periodic effort by phase or activity or person type (planned and actual) vs. time		(S)
Requirements		
5) Requirement count vs. time		(P)
6) Requirement verified vs. time		(P)
7) Requirement errors vs. time		(P)
8) Requirements satisfied by OTS (estimated)		(S)
9) Requirements satisfied by OTS (actual)		(S)
10) Number of ambiguous phrases		(S)
11) Number of optional phrases		(S)
12) Number of TBDs, TBSs, etc.		(S)
Modifications		
13) Number of modifications opened and closed vs. component/doc		(P)
14) Number of Modifications opened and closed (defined by program/project)		(P)
15) Effort staffed hours to perform modifications vs. component		(S)
16) Effort staffed hours to perform modifications vs. cause		(S)
Code		
17) Total source code size (estimated & actual) vs. time		(P)
18) Developed source code library size actual vs. time		(S)
19) Calling complexity by component		(S)
20) Cyclomatic complexity by component		(S)
21) Relative complexity by component		(S)
Defects		
22) Cumulative total number of open and closed defects over time		(P)
23) Cumulative number of open and closed defects by severity over time		(P)

Legend: P – Primary Metrics S – Secondary Metrics